REMARKS

Claims 1-55 were examined. Applicant has amended claims 1, 49 and 55 No claims are cancelled or are newly presented. No new matter has been introduced.

Amendments to the Drawings:

Figures 3 and 4 have been amended to include elements 32 and 56.

With regard to Figure 6, there is no element 54 in that Figure.

With regard to Figure 11, there is no element 381. There is an element 318.

Objections to the Specification

The specification is objected to because of formalities. Applicant has amended the specification. Paragraphs 42 has been amended, paragraphs 43-55 have been deleted. Applicant believes that changes are not needed.

Rejections under 35 U.S.C. §102

The examiner has rejected claims 1-5, 8-25 under §102(e) as being anticipated by Rosaria (US 6,976,246).

Claims 1, 11, 26-31, 33-34, 37, 39-55 stand rejected under §102(e) as anticipated by Kossatchev (US 6,698,012).

This ground of rejection is respectively traversed.

Rejections under 35 U.S.C. §103

Claims 2-4 stand rejected under §103(a) as being obvious over Rosaria (US 6,976,246).

Claims 32, 35-36 and 38 stand rejected under §103(a) as being obvious over Kossatchev (6,698,012).

This ground of rejection is respectively traversed.

In one embodiment of the present invention, as set forth in claim 1, a method is provided for generating test cases that are converted to an abstract representation. Semantic analysis is used to convert the test cases to abstract representations. A rule-based generation of test cases is provided from an abstract representation that includes

application states, external interaction sequences and input data of test cases from data stores. Each application state is a set of application objects associated with a set of attributes and their values, or represents a runtime snapshot of an application under test which defines a context of external interaction. Generated test cases are validated. The test cases are converted to test scripts.

Rosaria discloses a system testing interface that provides a user with an interface to develop and generate a model for a software application under test, initiate the development of a test sequence with a graph traversal algorithm, and initiate the generation of a test execution program to test the application.

Unlike the present invention, Rosaria does not convert test cases into an internal abstract representation independent of the tools. Rosaria does not use semantic analysis of scripts based on an application model.

In Rosaria, a system testing interface is provided with a graphical user interface and a model generation engine to implement a model-based testing tool. The user interface enables a user to define a state table and the associated software application transitions in terms of simple user-defined rules. From the user-defined rules, the model generation engine generates the entire model (i.e., state table) of the software application under test. The state table is generated when the model generation engine calculates and develops all of the possible operational mode value combinations for the software application, and for each permutation, determines which software application inputs are applicable.

After the model of the software application under test is generated, the testing interface enables a user to select a graph traversal program that generates a test sequence of inputs for the software application from the model. The testing interface also enables a user to initiate a test of the application by selecting a test driver program that executes the test sequence of inputs on the software application.

A rules editor enables a user to enter the parameters pertaining to transition information about the software application . The transition information includes a current state of the application which is associated with an input of the application, and a next state of the application. The next state of an application indicates the state of the application after the input has been applied to the current state of the application. A transition rule is defined by the current state, the input of the application, and the next

state. A graph traversal menu enables a user to select from among multiple graph traversal algorithms and to generate a test sequence of application inputs to test the software application in a particular order of application inputs. A test execution menu enables a user to select from among multiple test driver programs and to initiate a test of the software application.

As illustrated in Figure 1, Kossatchev et al. generates a verification system to verifying a procedure interface of a System Under Test (SUT). Unlike the present invention, Kossatcheve et al. does not convert test cases into an internal abstract representation independent of the tools. Kossatcheve et al. does not use semantic analysis of scripts based on an application model.

Parallel procedures are separated from consecutive procedures which have no parallelism; defining a group of parallel procedures to be tested in a parallel mode; specifying behaviour of the group of procedures in the parallel mode; and testing the group of procedures in the parallel mode separately from the consecutive procedures, based on the specified behaviour.

The verification system generator has a means for generating formal specifications, a test source generator and a repository. Based on the formal specifications, the test source generator generates test sources. The generated formal specifications and the test sources are stored in the repository.

The test sources are used to generate a test suite. The test suite is a set of programs and test data intended for the use in verifying the target procedure interface.

The formal specifications are generated in a form independent from implementation of the SUT. The test sources that are generated based on the implementation independent formal specification.

The generation of the verification system is carried out in two stages. First, generation of implementation independent programs is performed. Then, implementation independent programs are compiled into those in implementation language of the SUT .

Compiled test sources form the test suite. The test suite executes tests on the SUT and analyses results of the tests to verify the procedure interface.

A test case parameter generator generates test case parameter sources for generating test case parameters. That is, the test case parameter generator generates

constant arrays and programs that generate and select needed test case parameters. The test case parameters are represented by these constant arrays and programs.

Based on the formal specifications, the test driver generator generates test driver sources for generating test drivers. The test drivers execute tests on the SUT using the test case parameters in implementation environments and analysing results of tests.

The test drivers comprise programs to execute and control testing of the procedure interface. The test case parameters are parameters of a test case. A test case is an instance of a tested procedure. A test case is defined by a procedure name and its parameters, i.e., test case parameters. Also, state of environment may be a factor of defining a test case. The test drivers use the test case parameters and execute test cases on the SUT to verify the procedure interface.

The test driver generator generates the test driver sources which, once compiled into the test drivers by the implementation language compiler, fulfil functions to initialize the procedure interface, prepare input values, call tested procedures with test case parameters, and receive test procedure results and analysis of the test results. In general case, the test driver sources are complex programs.

CONCLUSION

It is submitted that the present application is in form for allowance, and such action is respectfully requested. The Commissioner is authorized to charge any additional fees which may be required, including petition fees and extension of time fees, to Deposit Account No. 08-1641 (Docket No. 07464-0004).

Respectfully submitted,
HELLER EHRMAN LLP

Date: 4·3·07

Paul Davis, Reg. No. 29,294

275 Middlefield Road
Menlo Park, CA 94205
(650) 324-7041
Customer No. 25213